Bike Renting

Pesala Ravi Kumar

14/08/2019

Contents

1. Introduction 2

1.1 Problem Statement 2

1.2 Data 2

2. Methodology 3

2.1 Pre Processing 3

2.1.1 Missing Value Analysis 3

2.1.2 Outlier Analysis 4

2.1.3 Feature Selection 4

2.2 Modeling 5

2.2.1 Model Selection 5

2.2.2 Linear Regression 7

3. Conclusion 10

3.1 Model Evaluation 10

3.1.1 MAPE 10

3.1.2 MSE & RMSE 10

Appendix 11

R code 11

Python code 17

Visualizations 22

**Chapter 1**

**Introduction**

**1.1 Problem Statement**

The objective of this Case is to Predication of bike rental count on daily based on the environmental and seasonal settings. Now a days there are many organizations who are running these bike rental work. In order to reduce the effort of renting the bike in different time periods and conditions by applying some machine learning concepts. We would like to predict the count of bike rental based on the conditions when the bike is renting by customers which are already known and easy to calculate further predictions.

**1.2 Data**

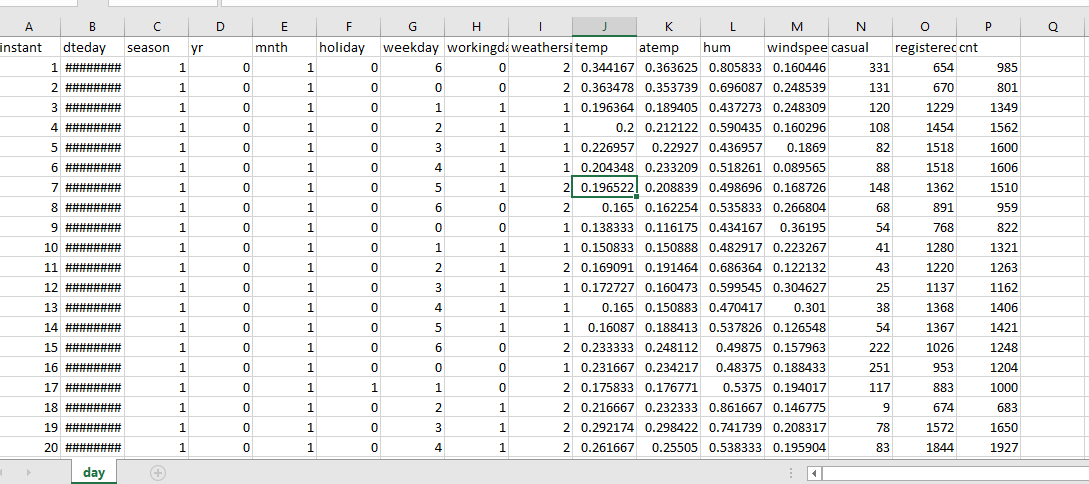


Fig 1.2.1 Data

As you can see in the fig.1.2.1 above we have the following 15 variables using which we have to predict the one of the variables.

1. Date
2. Season
3. Year
4. Month
5. Holiday
6. Weekday
7. Working Day
8. Weather
9. temp
10. atemp
11. Humidity
12. Casual
13. Registered
14. Count

**Chapter 2**

**Methodology**

**2.1 Pre Processing**

Any predictive modeling requires that we look at the data before we start modeling. However, in data mining terms looking at data refers to so much more than just looking. Looking at data refers to exploring the data, cleaning the data as well as visualizing the data through graphs and plots. This is often called as Exploratory Data Analysis.

**2.1.1 Missing Value Analysis**

In statistics, missing data, or missing values, occur when no data value is stored for the variable in an observation. Missing data are a common occurrence and can have a significant effect on the conclusions that can be drawn from the data.

Why missing the value?

Human Error, refuse to answer, optional

In order to handle Missing Values, there are 2 cases Ignore or Impute Missing value.

Before Imputing Understand why value is missing and by plotting graphs.

Delete the observations where you not to impute.

Techniques to impute missing values:

1. Fill with central statistics like
   * 1. Mean
     2. Mode
     3. Median
2. Distance base/ Data Mining Method – KNN Imputation
3. Prediction method – Machine Learning models

Selecting the technique for missing values

1. Create a small subset of total data.
2. Delete some values manually.
3. Use multiple methods to fill.
4. See which technique fills correctly.
5. Select that technique for finding missing value analysis.

**2.1.2 Outlier Analysis**

Outlier, it is an observation which inconsistence to rest of data.

Causes of the outlier are,

* 1. Poor data quality or contamination.
  2. Low quality measurements.
  3. Manual error.
  4. Malfunctioning equipment.
  5. Correct but exceptional data

Effect of outliers is that it will gives data which is not present in data because of the outlier.

Assume data as 1, 3, 5, 7, and 14 now try to impute using mean value. Mean will be 6 which is not present under data so it might reflect the modelling.

Steps to detect an outlier are,

1. Detect variable with outlier using graphical tools
2. Replace all with NA
3. Apply the missing value analysis on NA records to impute new Values.

**2.1.3 Feature Selection**

It is also called as variable selection or attribute selection.

Selecting a subset of relevant features like variables, predictors for model construction.

Advantages of Feature selection is Dimensionality Reduction (Variable reduction).

Techniques to dimensionality reduction,

1. Correlation analysis.
2. Chi-square test of independence.

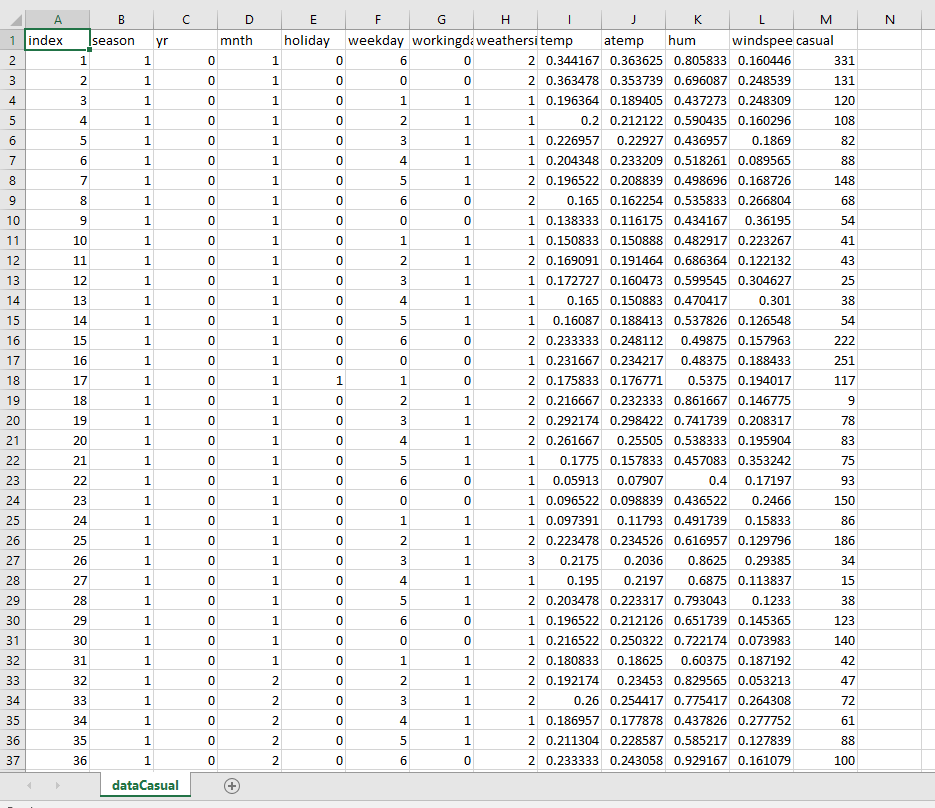
**2.2 Modeling**

**2.2.1 Model Selection**

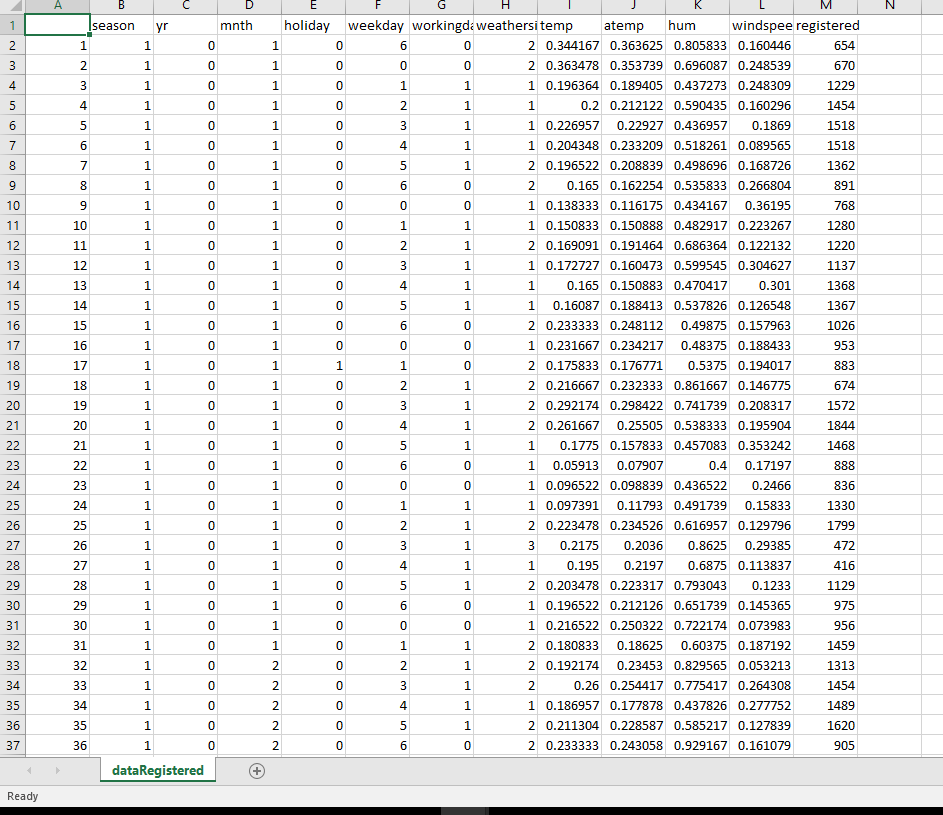
In our early stages of analysis during preprocessing we have come to understand that in order to predict the count of bike rentals we need to predict the casual and registered bikes so that we can add those two and calculate the total count of bike rentals.

In order to predict total count we’ll divide the data into 2 data sets with casual and registered variables separately.

Casual Dataset



Registered Dataset



You always start your model building from the simplest to more complex so after applying all the models select the best model according to accuracy and MAPE and etc.

**2.2.2 Linear Regression**

vif(saved\_dataCasual[,-12])

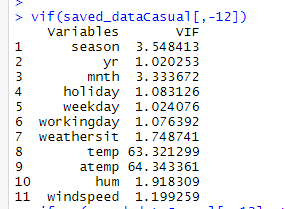


Fig 2.2.2.1 vif(Casualdata)

vifcor(saved\_dataCasual[,-12], th = 1.0)

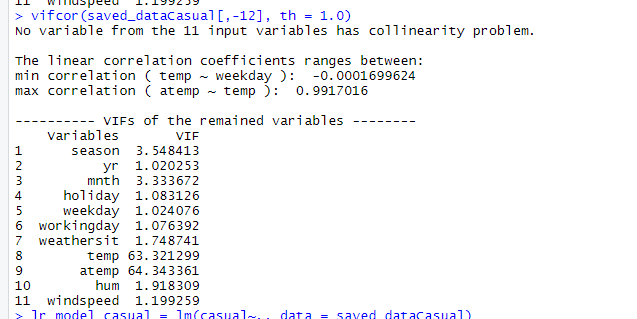


Fig 2.2.2.2 vifcor(CasualData)

lr\_model\_casual = lm(casual~., data = saved\_dataCasual)

summary(lr\_model\_casual)

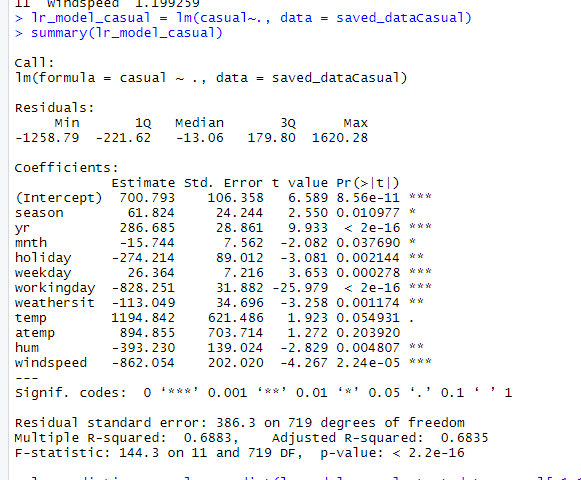


Fig 2.2.2.3 summary(linearRegressionModel)

vif(saved\_dataRegistered[,-12])

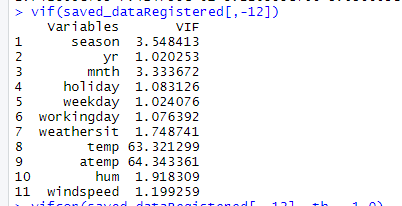


Fig 2.2.2.4 vif(RegisteredData)

vifcor(saved\_dataRegistered[,-12], th = 1.0)

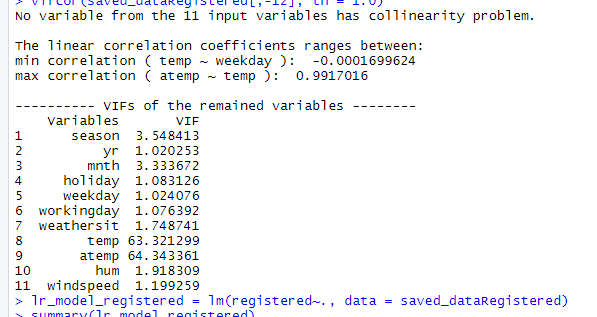


Fig 2.2.2.5 vifcor(RegisteredData)

lr\_model\_registered = lm(registered~., data = saved\_dataRegistered)

summary(lr\_model\_registered)

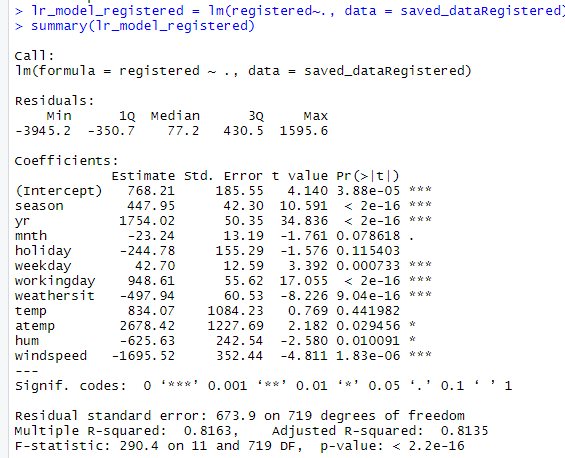


Fig 2.2.2.6 summary(linearRegressionModel)

**Chapter 3**

**Conclusion**

**3.1 Model Evaluation**

Now that we have a few models for predicting the target variable, we need to decide which one to choose. There are several criteria that exist for evaluating and comparing models. We can compare the models using any of the following criteria:

1. Predictive performance
2. MAPE
3. MSE
4. RMSE

Predictive performance can be measured by comparing Predictions of the models with real values of the target variables, and calculating some average error measure.

**3.1.1 MAPE (Mean Absolute Error)**

Measures accuracy as the percentage of error.

It is calculated as the average of the unsigned percentage error, as shown in the example below: Many organizations focus primarily on the MAPE when assessing forecast accuracy.

**3.1.2 MSE & RMSE (Mean Square Error & Root Mean Square Error)**

Steps to calculate MSE,

1. Find the regression line.
2. Insert your X values into the linear regression equation to find the new Y values
3. Subtract the new Y value from the original to get the error.
4. Square the errors.
5. Add up the errors.
6. Find the mean.

Calculate RMSE as root of MSE.

**Appendix**

**R code:**

rm(list=ls())

install.packages(c("dmm","dplyr","plyr","reshape","ggplot2","data.table","psych","usdm","caret","DMwR"))

data=read.csv("day.csv",header=T)

newData=subset(data, select = c("season","yr","mnth","holiday","weekday","workingday","weathersit","temp","atemp","hum","windspeed","casual","registered","cnt"))

savedData=newData

dataCasual = subset(newData, select = c("season","yr","mnth","holiday","weekday","workingday","weathersit","temp","atemp","hum","windspeed","casual"))

saved\_dataCasual = dataCasual;

dataRegistered = subset(newData, select = c("season","yr","mnth","holiday","weekday","workingday","weathersit","temp","atemp","hum","windspeed","registered"))

saved\_dataRegistered = dataRegistered

write.csv(dataCasual, "dataCasual.csv",row.names = T)

write.csv(dataRegistered, "dataRegistered.csv",row.names = T)

# Retrive numeric data

numeric\_index\_casual = sapply(dataCasual,is.numeric)

numeric\_data\_casual = dataCasual[,numeric\_index\_casual]

numeric\_data\_cols\_casual = colnames(numeric\_data\_casual)

numeric\_index\_registered = sapply(dataRegistered,is.numeric)

numeric\_data\_registered = dataRegistered[,numeric\_index\_registered]

numeric\_data\_cols\_registered = colnames(numeric\_data\_registered)

#Calculate outliers in dataCasual

dataCasual1 = dataCasual

for(i in numeric\_data\_cols\_casual)

{

print(i)

val\_casual = dataCasual1[,i][dataCasual1[,i]%in%boxplot.stats(dataCasual1[,i])$out]

print(length(val\_casual))

dataCasual1 = dataCasual1[which(!dataCasual1[,i]%in%val\_casual),]

}

# Replace all outliers in dataCasual with NA and impute missing using missing value analysis

dataCasual1 = dataCasual

for(i in numeric\_data\_cols\_casual)

{

val\_casual = dataCasual1[,i][dataCasual1[,i]%in%boxplot.stats(dataCasual1[,i])$out]

dataCasual1[,i][dataCasual1[,i]%in%val\_casual] = NA

}

#Calculate outliers in dataRegistered

dataRegistered1 = dataRegistered

for(i in numeric\_data\_cols\_registered)

{

print(i)

val\_registered = dataRegistered1[,i][dataRegistered1[,i]%in%boxplot.stats(dataRegistered1[,i])$out]

print(length(val\_registered))

dataRegistered1 = dataRegistered1[which(!dataRegistered1[,i]%in%val\_registered),]

}

# Replace all outliers in dataRegistered with NA and impute missing using missing value analysis

dataRegistered1 = dataRegistered

for(i in numeric\_data\_cols\_registered)

{

val\_registered=dataRegistered1[,i][dataRegistered1[,i]%in%boxplot.stats(dataRegistered1[,i])$out]

dataRegistered1[,i][dataRegistered1[,i]%in%val\_registered] = NA

}

# Apply KNN imputation for dataCasual

require(DMwR)

dataCasual1 = knnImputation(dataCasual1,k=5)

# Apply Mean imputation for columns with NA (because of error: there are not sufficient cases)

dataCasual1$holiday[is.na(dataCasual1$holiday)] = mean(dataCasual1$holiday,na.rm = T)

dataCasual1$hum[is.na(dataCasual1$hum)] = mean(dataCasual1$hum,na.rm = T)

dataCasual1$windspeed[is.na(dataCasual1$windspeed)] = mean(dataCasual1$windspeed,na.rm = T)

dataCasual1$casual[is.na(dataCasual1$casual)] = mean(dataCasual1$casual,na.rm = T)

# Apply KNN imputation for dataRegistered

dataRegistered1 = knnImputation(dataRegistered1,k=5)

# Apply Mean imputation for columns with NA (because of error: there are not sufficient cases)

dataRegistered1$holiday[is.na(dataRegistered1$holiday)] = mean(dataRegistered1$holiday,na.rm = T)

dataRegistered1$hum[is.na(dataRegistered1$hum)] = mean(dataRegistered1$hum,na.rm = T)

dataRegistered1$windspeed[is.na(dataRegistered1$windspeed)] = mean(dataRegistered1$windspeed,na.rm = T)

dataRegistered1$registered[is.na(dataRegistered1$registered)] = mean(dataRegistered1$registered,na.rm = T)

# chi-square test of Independency

# factor\_index=sapply(dataCasual1, is.factor)

# factor\_data=dataCasual1[,factor\_index]

#

# for (i in 1:ncol(dataCasual1)-1)

# {

# print(chisq.test(table(dataCasual1$dataCasual1[,length(dataCasual1)],dataCasual1[,i])))

# }

# select Dependent columns from dataCasual

nonDependentCols\_casual = names(dataCasual1)%in% c("casual")

dependentData\_casual = dataCasual1[!nonDependentCols\_casual]

dependentCols\_casual = colnames(dependentData\_casual)

# select Dependent columns from dataRegistered

nonDependentCols\_registered = names(dataRegistered1)%in% c("registered")

dependentData\_registered = dataRegistered1[!nonDependentCols\_registered]

depedentCols\_registered = colnames(dependentData\_registered)

# Feature Scaling

# require(graphics)

# for(i in dependentCols\_casual)

# {

# print(i)

# qqnorm(dependentData\_casual$i)

# hist(dependentData\_casual$i)

# }

# Sampling Techniques

d\_dataCasual = dependentData\_casual

simpleRandomSampling\_dataCasual = d\_dataCasual[sample(nrow(d\_dataCasual),100,replace = F),]

d\_dataRegistered = dependentData\_registered

simpleRandomSampling\_dataRegistered = d\_dataRegistered[sample(nrow(d\_dataRegistered),100,replace = F),]

# # divide train & test data

train\_index\_casual = sample(1:nrow(saved\_dataCasual), 0.8 \* nrow(saved\_dataCasual), prob = NULL)

train\_data\_casual = saved\_dataCasual[train\_index\_casual,]

test\_data\_casual = saved\_dataCasual[-train\_index\_casual,]

train\_index\_registered = sample(1:nrow(saved\_dataRegistered), 0.8 \* nrow(saved\_dataRegistered), prob = NULL)

train\_data\_registered = saved\_dataRegistered[train\_index\_registered,]

test\_data\_registered = saved\_dataRegistered[-train\_index\_registered,]

# # builds decision tree

# library(rpart)

# fit = rpart(registered~., data=train\_data\_registered, method="anova")

# library(MASS)

# predictions = predict(fit, test\_data\_registered[,-12])

# # Calculate MAPE, MSE, RMSE, MAE

# library(DMwR)

# regr.eval(test\_data\_registered[,12], predictions, stats = c('mae','mape','mse','rmse'))

# mae mape mse rmse

# 5.827114e+02 2.279066e+00 7.503266e+05 8.662140e+02

# KNN

# install.packages("caret")

# train\_index\_casual = sample(1:nrow(saved\_dataCasual), 0.8 \* nrow(saved\_dataCasual), prob = NULL)

# train\_data\_casual = saved\_dataCasual[train\_index\_casual,]

# test\_data\_casual = saved\_dataCasual[-train\_index\_casual,]

# train\_index\_registered = sample(1:nrow(saved\_dataRegistered), 0.8 \* nrow(saved\_dataRegistered), prob = NULL)

# train\_data\_registered = saved\_dataRegistered[train\_index\_registered,]

# test\_data\_registered = saved\_dataRegistered[-train\_index\_registered,]

# library(class)

# knn\_predictions\_casual = knn(train\_data\_casual[,1:11], test\_data\_casual[,1:11], train\_data\_casual$casual, k=5)

# knn\_predictions\_registered = knn(train\_data\_registered[,1:11], test\_data\_registered[,1:11], train\_data\_registered$registered, k=5)

# #Accuracy

# knn\_CM\_casual = table(knn\_predictions\_casual, test\_data\_casual$casual)

# sum(diag(knn\_CM\_casual))/nrow(test\_data\_casual)

# TN\_casual = knn\_CM\_casual[0,0]

# FN\_casual = knn\_CM\_casual[1,0]

# TP\_casual = knn\_CM\_casual[1,1]

# FP\_casual = knn\_CM\_casual[0,1]

#

# knn\_CM\_registered = table(knn\_predictions\_registered, test\_data\_registered$registered)

# sum(diag(knn\_CM\_registered))/nrow(test\_data\_registered)

# TN\_registered = knn\_CM\_registered[0,0]

# FN\_registered = knn\_CM\_registered[1,0]

# TP\_registered = knn\_CM\_registered[1,1]

# FP\_registered = knn\_CM\_registered[0,1]

#

# library(DMwR)

# regr.eval(test\_data\_casual[,12], knn\_predictions\_casual, stats = c('mae','mape','mse','rmse'))

# regr.eval(test\_data\_registered[,12], knn\_predictions\_registered, stats = c('mae','mape','mse','rmse'))

# Linear Regression (Registered)

install.packages("usdm")

library(usdm)

vif(saved\_dataCasual[,-12])

vifcor(saved\_dataCasual[,-12], th = 1.0)

lr\_model\_casual = lm(casual~., data = saved\_dataCasual)

summary(lr\_model\_casual)

lr\_prediction\_casual = predict(lr\_model\_casual, test\_data\_casual[,1:11])

library(DMwR)

library(MASS)

regr.eval(test\_data\_registered[,12], lr\_prediction\_casual, stats = c('mae','mape','mse','rmse'))

vif(saved\_dataRegistered[,-12])

vifcor(saved\_dataRegistered[,-12], th = 1.0)

lr\_model\_registered = lm(registered~., data = saved\_dataRegistered)

summary(lr\_model\_registered)

lr\_prediction\_registered = predict(lr\_model\_registered, test\_data\_casual[,1:11])

library(DMwR)

library(MASS)

regr.eval(test\_data\_registered[,12], lr\_prediction\_registered, stats = c('mae','mape','mse','rmse'))

# KMeans Clustering

install.packages("NbClust")

library(NbClust)

d\_casual = saved\_dataCasual

clusters\_casual = NbClust(d\_casual, min.nc = 2, max.nc = 10, method = "kmeans")

barplot(table(clusters\_casual$Best.nc[1,]), xlab="X", ylab="Y", main="")

kmeans\_model\_casual = kmeans(d\_casual,4,nstart = 25)

cluster\_accuracy\_casual = table(d\_casual$casual,kmeans\_model\_casual$cluster)

d\_registered = saved\_dataRegistered

clusters\_registered = NbClust(d\_registered, min.nc = 2, max.nc = 10, method = "kmeans")

barplot(table(clusters\_registered$Best.nc[1,]), xlab="X", ylab="Y", main="")

kmeans\_model\_registered = kmeans(d\_registered,4,nstart = 25)

cluster\_accuracy\_registered = table(d\_registered$registered,kmeans\_model\_registered$cluster)

library(ggplot2)

library(scales)

library(psych)

library(gplots)

newData\_casual = dataCasual

newData\_registered = dataRegistered

# Bar plot ( Categorical variables VS Target variable)

# Casual Data

ggplot(newData\_casual, aes\_string(x=newData\_casual$season, y=newData\_casual$casual)) +

geom\_bar(stat = "identity",fill="Blue") + theme\_bw() +

xlab("season") + ylab("casual") +

scale\_y\_continuous(breaks = pretty\_breaks(n=10)) +

ggtitle("Bar plot Season vs Casual ") + theme(text = element\_text(size=10))

ggplot(newData\_casual, aes\_string(x=newData\_casual$mnth, y=newData\_casual$casual)) +

geom\_bar(stat = "identity",fill="Blue") + theme\_bw() +

xlab("month") + ylab("casual") +

scale\_y\_continuous(breaks = pretty\_breaks(n=10)) +

ggtitle("Bar plot Month vs Casual ") + theme(text = element\_text(size=10))

ggplot(newData\_casual, aes\_string(x=newData\_casual$holiday, y=newData\_casual$casual)) +

geom\_bar(stat = "identity",fill="Blue") + theme\_bw() +

xlab("holiday") + ylab("casual") +

scale\_y\_continuous(breaks = pretty\_breaks(n=10)) +

ggtitle("Bar plot Holiday vs Casual ") + theme(text = element\_text(size=10))

ggplot(newData\_casual, aes\_string(x=newData\_casual$weekday, y=newData\_casual$casual)) +

geom\_bar(stat = "identity",fill="Blue") + theme\_bw() +

xlab("weekday") + ylab("casual") +

scale\_y\_continuous(breaks = pretty\_breaks(n=10)) +

ggtitle("Bar plot Weekday vs Casul ") + theme(text = element\_text(size=10))

ggplot(newData\_casual, aes\_string(x=newData\_casual$workingday, y=newData\_casual$casual)) +

geom\_bar(stat = "identity",fill="Blue") + theme\_bw() +

xlab("workingday") + ylab("casual") +

scale\_y\_continuous(breaks = pretty\_breaks(n=10)) +

ggtitle("Bar plot Working Day vs Casual") + theme(text = element\_text(size=10))

ggplot(newData\_casual, aes\_string(x=newData\_casual$weathersit, y=newData\_casual$casual)) +

geom\_bar(stat = "identity",fill="Blue") + theme\_bw() +

xlab("Weather") + ylab("casual") +

scale\_y\_continuous(breaks = pretty\_breaks(n=10)) +

ggtitle("Bar plot weather vs casual ") + theme(text = element\_text(size=10))

# Registered Data

ggplot(newData\_registered, aes\_string(x=newData\_registered$season, y=newData\_registered$registered)) +

geom\_bar(stat = "identity",fill="Blue") + theme\_bw() +

xlab("season") + ylab("casual") +

scale\_y\_continuous(breaks = pretty\_breaks(n=10)) +

ggtitle("Bar plot Season vs Registered ") + theme(text = element\_text(size=10))

ggplot(newData\_registered, aes\_string(x=newData\_registered$mnth, y=newData\_registered$registered)) +

geom\_bar(stat = "identity",fill="Blue") + theme\_bw() +

xlab("month") + ylab("casual") +

scale\_y\_continuous(breaks = pretty\_breaks(n=10)) +

ggtitle("Bar plot Month vs Registered ") + theme(text = element\_text(size=10))

ggplot(newData\_registered, aes\_string(x=newData\_registered$holiday, y=newData\_registered$registered)) +

geom\_bar(stat = "identity",fill="Blue") + theme\_bw() +

xlab("holiday") + ylab("casual") +

scale\_y\_continuous(breaks = pretty\_breaks(n=10)) +

ggtitle("Bar plot Holiday vs Registered ") + theme(text = element\_text(size=10))

ggplot(newData\_casual, aes\_string(x=newData\_casual$weekday, y=newData\_casual$casual)) +

geom\_bar(stat = "identity",fill="Blue") + theme\_bw() +

xlab("weekday") + ylab("casual") +

scale\_y\_continuous(breaks = pretty\_breaks(n=10)) +

ggtitle("Bar plot Weekday vs Registered ") + theme(text = element\_text(size=10))

ggplot(newData\_registered, aes\_string(x=newData\_registered$workingday, y=newData\_registered$registered)) +

geom\_bar(stat = "identity",fill="Blue") + theme\_bw() +

xlab("workingday") + ylab("casual") +

scale\_y\_continuous(breaks = pretty\_breaks(n=10)) +

ggtitle("Bar plot Working Day vs Registered") + theme(text = element\_text(size=10))

ggplot(newData\_registered, aes\_string(x=newData\_registered$weathersit, y=newData\_registered$registered)) +

geom\_bar(stat = "identity",fill="Blue") + theme\_bw() +

xlab("Weather") + ylab("casual") +

scale\_y\_continuous(breaks = pretty\_breaks(n=10)) +

ggtitle("Bar plot Weather vs Registered ") + theme(text = element\_text(size=10))

**Python code:**

import os

os.getcwd()

os.chdir("C:/Users/gopin/Documents/R/BikeRental-Project")

import pandas as pd

import numpy as np

import matplotlib as mlt

import matplotlib.pyplot as plt

from sklearn.neighbors import KNeighborsClassifier

data = pd.read\_csv("day.csv")

savedData = data

dataCasual = savedData[["season","yr","mnth","holiday","weekday","workingday","weathersit","temp","atemp","hum","windspeed","casual"]]

dataRegsitered = savedData[["season","yr","mnth","holiday","weekday","workingday","weathersit","temp","atemp","hum","windspeed","registered"]]

dCasual = dataCasual.copy()

dRegistered = dataRegsitered.copy()

# Store continuous variable names

cnames\_C = ["season","yr","mnth","holiday","weekday","workingday","weathersit","temp","atemp","hum","windspeed","casual"]

cnames\_R = ["season","yr","mnth","holiday","weekday","workingday","weathersit","temp","atemp","hum","windspeed","registered"]

# Detect outliers & delete(Casual)

for i in cnames\_C:

q75, q25 = np.percentile(dCasual.loc[:,i],[75,25])

iqr = q75 - q25

innerfence = q25 - (iqr \* 1.5)

outerfence = q75 + (iqr \* 1.5)

dCasual = dCasual.drop(dCasual[dCasual.loc[:,i] < innerfence].index)

dCasual = dCasual.drop(dCasual[dCasual.loc[:,i] > outerfence].index)

# Replace with NA

dCasual = dataCasual.copy()

for i in cnames\_C:

q75, q25 = np.percentile(dCasual.loc[:,i],[75,25])

iqr = q75 - q25

innerfence = q25 - (iqr \* 1.5)

outerfence = q75 + (iqr \* 1.5)

dCasual.loc[dCasual[i] < innerfence,:i] = np.nan

dCasual.loc[dCasual[i] > outerfence,:i] = np.nan

# Calculate Missing values

missing\_C = pd.DataFrame(dCasual.isnull().sum())

# Impute using Mode method

for i in cnames\_C:

dCasual[i] = dCasual[i].fillna(dCasual[i].mode()[0])

#missing\_C = pd.DataFrame(dCasual.isnull().sum())

savedDataCasual = dCasual

# Detect outliers & delete (Registered)

for i in cnames\_R:

q75, q25 = np.percentile(dRegistered.loc[:,i],[75,25])

iqr = q75 - q25

innerfence = q25 - (iqr \* 1.5)

outerfence = q75 + (iqr \* 1.5)

dRegistered = dRegistered.drop(dRegistered[dRegistered.loc[:,i] < innerfence].index)

dRegistered = dRegistered.drop(dRegistered[dRegistered.loc[:,i] > outerfence].index)

# Replace with NA

dRegistered = dRegistered.copy()

for i in cnames\_R:

q75, q25 = np.percentile(dRegistered.loc[:,i],[75,25])

iqr = q75 - q25

innerfence = q25 - (iqr \* 1.5)

outerfence = q75 + (iqr \* 1.5)

dRegistered.loc[dRegistered[i] < innerfence,:i] = np.nan

dRegistered.loc[dRegistered[i] > outerfence,:i] = np.nan

# Calculate Missing values

missing\_R = pd.DataFrame(dRegistered.isnull().sum())

# Impute using Mode method

for i in cnames\_R:

dRegistered[i] = dRegistered[i].fillna(dRegistered[i].mode()[0])

savedDataRegistered = dRegistered

# Feature selection

'''import seaborn as sns

from scipy.stats import chi2\_contingency

from random import randrange,uniform

for i in dCasual.columns:

print(i)

p = chi2\_contingency(pd.crosstab(dCasual['casual'],dCasual[i]))

'''

# Gives Barplot

# %matplotlib inline

# plt.hist(dCasual['weekday'], bins='auto')

# Sampling using Systematic sampling

simpleRandomSampling\_C = dCasual.sample(100)

simpleRandomSampling\_R = dRegistered.sample(100)

from sklearn import tree

from sklearn.metrics import accuracy\_score

from sklearn.model\_selection import train\_test\_split

# Train and Test data

dCasual = savedDataCasual.copy()

xc = dCasual.values[:, 0:11]

yc = dCasual.values[:, 11]

xc\_train, xc\_test, yc\_train, yc\_test = train\_test\_split(xc,yc,test\_size=0.2)

# Linear Regression model for Data Casual

import statsmodels.api as sm

train\_c, test\_c = train\_test\_split(dCasual,test\_size=0.2)

model\_C = sm.OLS(train\_c.iloc[:,11], train\_c.iloc[:,0:11]).fit()

model\_C.summary()

dRegistered = savedDataRegistered.copy()

xr = dRegistered.values[:, 0:11]

yr = dRegistered.values[:, 11]

xr\_train, xr\_test, yr\_train, yr\_test = train\_test\_split(xr,yr,test\_size=0.2)

# Linear Regression model for Data Registered

import statsmodels.api as sm

train\_c, test\_c = train\_test\_split(dRegistered,test\_size=0.2)

model\_R = sm.OLS(train\_c.iloc[:,11], train\_c.iloc[:,0:11]).fit()

model\_R.summary()

import ggplot

from ggplot import \*

dC = savedDataCasual.copy()

ggplot(dC, aes(x='season', y='casual')) +\

geom\_bar(fill="blue") +\

scale\_color\_brewer(type="diverging", palette=4) +\

xlab("Season") + ylab("Casual") + ggtitle("Barplot\_seasonVScasual") + theme.bw()

ggplot(dC, aes(x='holiday', y='casual')) +\

geom\_bar(fill="blue") +\

scale\_color\_brewer(type="diverging", palette=4) +\

xlab("Holiday") + ylab("Casual") + ggtitle("Barplot\_holidayVScasual") + theme.bw()

ggplot(dC, aes(x='mnth', y='casual')) +\

geom\_bar(fill="blue") +\

scale\_color\_brewer(type="diverging", palette=4) +\

xlab("Month") + ylab("Casual") + ggtitle("Barplot\_monthVScasual") + theme.bw()

ggplot(dC, aes(x='weather', y='casual')) +\

geom\_bar(fill="blue") +\

scale\_color\_brewer(type="diverging", palette=4) +\

xlab("Weather") + ylab("Casual") + ggtitle("Barplot\_weatherVScasual") + theme.bw()

ggplot(dC, aes(x='weekday', y='casual')) +\

geom\_bar(fill="blue") +\

scale\_color\_brewer(type="diverging", palette=4) +\

xlab("Weekday") + ylab("Casual") + ggtitle("Barplot\_weekDayVScasual") + theme.bw()

ggplot(dC, aes(x='workkingday', y='casual')) +\

geom\_bar(fill="blue") +\

scale\_color\_brewer(type="diverging", palette=4) +\

xlab("Working Day") + ylab("Casual") + ggtitle("Barplot\_workingDayVScasual") + theme.bw()

ggplot(dC, aes(x='season', y='casual')) +\

geom\_bar(fill="blue") +\

scale\_color\_brewer(type="diverging", palette=4) +\

xlab("Season") + ylab("Casual") + ggtitle("Barplot\_seasonVScasual") + theme.bw()

dR = savedDataRegistered.copy()

ggplot(dR, aes(x='season', y='registered')) +\

geom\_bar(fill="blue") +\

scale\_color\_brewer(type="diverging", palette=4) +\

xlab("Season") + ylab("Registered") + ggtitle("Barplot\_seasonVSregistered") + theme.bw()

ggplot(dR, aes(x='holiday', y='casual')) +\

geom\_bar(fill="blue") +\

scale\_color\_brewer(type="diverging", palette=4) +\

xlab("Holiday") + ylab("Registered") + ggtitle("Barplot\_holidayVSregistered") + theme.bw()

ggplot(dR, aes(x='mnth', y='casual')) +\

geom\_bar(fill="blue") +\

scale\_color\_brewer(type="diverging", palette=4) +\

xlab("Month") + ylab("Registered") + ggtitle("Barplot\_monthVSregistered") + theme.bw()

ggplot(dR, aes(x='weather', y='casual')) +\

geom\_bar(fill="blue") +\

scale\_color\_brewer(type="diverging", palette=4) +\

xlab("Weather") + ylab("Registered") + ggtitle("Barplot\_weatherVSregistered") + theme.bw()

ggplot(dR, aes(x='weekday', y='casual')) +\

geom\_bar(fill="blue") +\

scale\_color\_brewer(type="diverging", palette=4) +\

xlab("Weekday") + ylab("Registered") + ggtitle("Barplot\_weekDayVSregistered") + theme.bw()

ggplot(dR, aes(x='workkingday', y='casual')) +\

geom\_bar(fill="blue") +\

scale\_color\_brewer(type="diverging", palette=4) +\

xlab("Working Day") + ylab("Registered") + ggtitle("Barplot\_workingDayVSregistered") + theme.bw()

**Visualizations:**

